Problem solving to teach advanced algorithms in heterogeneous groups

Florent Bouchez-Tichadou

Problem Solving to Teach Advanced Algorithms in Heterogeneous Groups

Florent Bouchez-Tichadou
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG
Grenoble, France
florent.bouchez-tichadou@imag.fr

ABSTRACT

It is notoriously difficult to teach heterogeneous groups. In this article I describe my experience in teaching high-level algorithms to a class of international master students coming from very different backgrounds, how the course evolved in a period spanning five years from a very classical course to a course that focuses highly on solving algorithmic problems in groups, and how that helped in better choosing the themes covered in the course, structuring it using a more definite direction, and better engaging almost all students by making them work on problems they can relate to.

This article presents the various axes I chose to follow as well as what was discarded from the initial course. It shows that students are more engaged in learning and more motivated to work, as there is the possibility for everyone to contribute and students can help one another; The course has then become a favorite amongst the other courses of the curriculum. Moreover, the teachers themselves are also more engaged, feel closer to the students, but are also under more pressure as the position changes dramatically from that of a lecturer. This article shows that it takes a lot of time to stabilize to satisfaction and presents recommendations for teachers interested in modifying their courses.

CCS CONCEPTS

• **Social and professional topics** → **Computer science education**; *Computational thinking*; Adult education;

KEYWORDS

Problem-Based Learning, Algorithms, Master level, Heterogeneous.

ACM Reference Format:
Florent Bouchez-Tichadou. 2018. Problem Solving to Teach Advanced Algorithms in Heterogeneous Groups. In *Proceedings of 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'18).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3197091.3197147

1 INTRODUCTION

When I started working at the university I got in charge of a master level course on advanced algorithms in an international master program. Students come from many different places around the world: mainly Europe, but also Asia, Middle East, and some from Africa or America. The main group, about 30%, come from our own country: France. Although there is a selection, it is very hard to judge precisely the capabilities and knowledge of all those people based on their applications. They are expected to have the equivalent of the European L3 degree in computer science, but what they have actually studied before is highly variable. As a consequence, the population in this master program is very heterogeneous.

The first year, I followed the same outline as my predecessor, introducing all topics as best as I could in a mixture of lectures and exercises: complexity analysis, amortized complexity, advanced data structures such as auto-balancing trees, dynamic programming, NP-completeness, approximation algorithms, and graph coloring. The students' competences in algorithms were very varied, so I tried my best to explain the basic knowledge while also trying to keep the interest of students who performed better. By the end of the semester, the morale was pretty low, for me, and seemed not much better for the majority of students: actually, I could not really tell for sure, as my feeling was that *I did not really know them.*

Here started my journey: I had to change something, for the way I was teaching felt wrong to me. I discovered *Problem-Based Learning* (PBL) [2, 5, 6], which seemed to promise to solve all the problem I had: students would work more during sessions, those with weaknesses in algorithms would catch up during those times, students would be more interested and invested when confronted to real-life problems. Problem-based learning is used in a variety of context and curricula, and in computer science, one can find many instances where it is used for first year students just starting to learn programming [2, 3, 5]. Although the principle applied is similar, there are some specificities when using it for teaching algorithms at the master level that I will present in Section 2.

The content of the course was very dense, and making students think more by themselves takes more time than just giving them new knowledge. Section 3 presents the choices that were made in terms of scientific content, but also how this reduction necessity forced me to better frame the essential and to better organize the course around what I view as the most important notions: the optimal and complexity properties of algorithms. The students' feedback for this course was recorded every year with a thorough survey. Analysis of these surveys is presented in section 4.

Finally, I will give on overview on the perceived effect of PBL on students, but also a return of experience from the *tutor*'s point of view, as this is the role that teachers endorse for this course. Section 5 presents the changes I experienced and in my relations with my students, as well as the effects on the other teachers who participated in this journey.

2 THE SET-UP

The course spans over 22 sessions of 1.5h. Problems require an average of 5-6 sessions since they are used to introduce difficult concepts, meaning there is enough space for four problems.

ITiCSE'18, July 2–4, 2018, Larnaca, Cyprus
2018. https://doi.org/10.1145/3197091.3197147

2.1 General Framing for a Problem

A *problem* spans about 5-6 sessions. The first session is devoted to the discovery of the problem, and a document of about 6 to 8 pages is distributed to all students. Its structure is as follows:

- The global learning objectives: to give students a general idea of what they will learn when trying to solve this problem. For instance: "using dynamic programming to find the optimal solution of a problem in polynomial time";
- The general session organization: the approximate milestones that we expect them to achieve at each session, and the work we expect from them between sessions;
- The problem itself: it introduces a "real-life problem" exposed in the everyday language, to that any modeling has to be done by the students themselves. See Appendix A for an full example of a problem used to introduce the concept of dynamic programming;
- A list of resources where students can search for the scientific knowledge they are currently lacking to solve the problem. These include "lecture notes," reference to chapters in books, or websites covering the topic;
- A more detailed planning of the sessions, especially for the first problems, to guide students in the steps to follow to try and solve the problem. By the end of the semester, this is not required as all groups know how to organize themselves;
- A "Your Learning" part listing precise learning objectives where students can review whether they have missed something in their learning. Items are for instance "writing a greedy algorithm to solve a problem" or "proving that a greedy algorithm is not optimal". After each problem, all students are individually evaluated on those items in a 30-40 minutes "quick test";
- Finally, a meta-analysis section where, after each problem, students are invited to reflect on their work (as a group and individually) during this problem. In particular it is important for them to reflect on what worked, what did not, and what they would like to change to improve for the next problem.

During the first session of a problem, each student is encouraged to first read the document and start thinking *alone* on the problem. This stage is particularly important as we want first to make students think *by themselves* and not rely purely on the group. When everyone has had enough time to start having their own ideas, they are then encouraged to share their comprehension of the problem. This part opens discussions and preconceptions are put to the test. After a few sessions, there is a natural equilibrium in every group, where some time is devoted to personal work and some other to sharing and discussion.

Before the end of the first session, each group must decide what needs to be done *individually* to prepare the next session. At the beginning, we suggest the work (reading, devising an algorithm…), but as the semester progresses they are autonomous. So between each session, the *same personal work* is expected from all students, so that the next session starts with the sharing and confrontation of personal ideas. Whenever a group feels they have attained a milestone, they are free to continue working on the next one.

Finally, students are required to hand out a group report on the problem at the end of the last session. This plays a very important role, as students usually have a very good idea on how to solve the problem and which algorithms to use, but they have poor communication skills with regard to actually explaining to someone else how to do it. They are expected to write algorithms in pseudo-code as well as all that would be required for a person to understand how to solve the problem. We want them to use pseudo-code as a means to communicate between *human beings* (contrary to actual code, which are instructions to a machine).[1]

2.2 Particular Sessions

The very first session of the semester. It is used to introduce the PBL method. Groups of students of 5-6 people are formed (see Section 2.5 for details) and are asked to solve a very small problem under time constraints. We use a projector to run a binary counter counting down to 0 that changes only one bit at a time. Teams of students compete to guess the time at which it will stop. The situation is fun and engaging and serves both as an ice-breaker (many students here are just arriving from their home country), and to show it is more efficient to work in a group than individually. This also allows us to cover a small topic that did not fit anymore in our problem organization: amortized complexity.

Restructuring lectures. Twice in the semester, for the problems introducing dynamic programming and approximation algorithms, we have a *restructuring lecture*. It is always the fourth session of a 6-session problem, which means that students will have already thought about the problem for three sessions, ruling out simple algorithms and starting to apply concepts found in the lecture notes. This is a particularly ripe time to give a "lecture" on the topic at hand, in the form of a Q&A session. Students are encouraged to ask questions in advance so they have to work out what are the obstacles that block their understanding. They are then very receptive and hungry for answers during that session.

2.3 The Tutor's Role

In PBL, the teacher becomes a *tutor*, trying to avoid as much as possible the role of the "one who knows." Initially, the students will want to check with the tutor if what they are doing is correct. What we want is to habituate them to be more autonomous, as in their future life they will not magically have someone to check whether they are doing right or wrong. Our role here is to ensure that learning takes place. As such, we keep track of the advancement of each group by discussing with them regularly (at least once but often more per session), questioning them to verify that all members really understand what is being discussed. We also have the role of "safety check," meaning that if we realize a group is really on the wrong track and will waste too much time we try to steer them in the right direction, often through the use of questions. A major difficulty is refraining ourselves from going back to a teacher position that would provide knowledge and answers to ensure all groups solve the problem.

2.4 Evaluation

This course targets *individual learning* through the use of *group work*. This means the emphasis in the grade will be on individual

[1] Although actually programming the algorithms is encouraged, we do not enforce it as students already do a solid amount of programming in the other courses of the master program.

evaluation, but we do not want people to disengage from the group work so a small portion of the grade is based on the quality of the group production.[2] Each problem is followed by a "quick" personal test of 30 to 40 minutes to encourage students to work continuously during the semester, and to check regularly their comprehension of the topics covered by the problems. 70% of the overall grade is based on a 3h final examination at the end of the semester; 20% on the 3 best quick tests (over 4), and 10% on the group reports.

2.5 Forming the Groups

We let them choose their groups of 5-6 people however they want, with the constraint that no more than two people speaks the same language, to encourage mixing up the origins. I initially also wanted that no female is left alone in a group, but dropped this constraint as this was never a problem at this level of study (female students seem accustomed to working with and handling male students, and the distribution of role, parole, and work was fair).[3] The number of students per group is chosen so that group work is possible (> 3 people) but groups are still manageable (< 7 people).

3 CONTENT

The content of the course prior to the use of PBL was dense, featuring complexity analysis, amortized complexity, auto-balancing trees, dynamic programming, NP-completeness, approximation algorithms, and graph coloring. During my first year of teaching, I had the impression of rushing through every topic. Looking back at that time, it feels like I was force-feeding a solid percentage of the students.

I realized that it was impossible to keep everything while switching to PBL. I also realized that some topics were just too difficult for students: either we would systematically avoid them in the final exam, or students would fail those questions, including those considered the "best" students. This forced me into rethinking what is important for students to understand at this level of study.

3.1 The Algorithmic Thinking at Master's Level

After four years of adjusting the course in PBL fashion, I have extracted what is now my guide when teaching algorithms at this level. The most important aspect is that students need to be able to reason on the following two characteristics: • the *optimality* of an algorithm; • and the *complexity* of an algorithm.

When students enter this course, they have some preconceptions on these notions, which highly differ from the scientific definitions. They will nearly always mistake one for the other, the most common misconception being that an algorithm is not "optimal" when we can devise a faster one for the same problem… This last point shouldn't be surprising, as for nearly all their studies, every algorithm encountered was optimal, and every problem had an optimal algorithm that solves it. I then organized the problems with these two notions as a thread of continuity, with the goal of making students think, analyse, devise, and write algorithms that vary a lot in complexity and optimality.

3.2 The Four Problems

The four problems we use try to induce the following logical steps: 1) A problem is presented along with a first algorithm, say Algorithm N, that "solves" it (a naïve or greedy algorithm). 2) Students realize that Algorithm N is actually not optimal and search for a better one: Algorithm B which may or may not be optimal. 3) These steps are repeated until they found optimal Algorithm E. 4) Students realize Algorithm E has exponential complexity and is not useable in practice. With the help of the lecture notes, students proceed to find polynomial Algorithm X, which either uses dynamic programming or is an α-approximation.

One important point here is the repetition throughout all the problems of the same steps. Students are so used to being guided step by step with a much finer grain that they are at loss when encountering a bigger, open problem. The steps presented above are a "meta-algorithm" that can be followed when solving algorithmic problems by asking the right questions: "is my algorithm optimal?", "can I find counter examples?", "is my algorithm exponential?", "is it an approximation?", …

I will now briefly present the four problems, explaining how they fit the above goal of giving student a profound understanding of the optimality and complexity of the algorithms, while also covering interesting algorithmic topics.

Problem 1: Maze generation. Actually a "warm-up" problem that introduces PBL, hence follows only loosely the above method. Students are given an informal algorithm that recursively generates a maze, and are asked first to translate the algorithm into precise pseudo-code, then propose solutions to escape the maze.

The beginning of the semester can be challenging for international students. They need some time to acclimate, and sometimes lack basic algorithmic knowledge. So this short problem only uses what every student is supposed to know at this level, but is still challenging enough so that all students can exercise their algorithms skills: while building the maze is easy, there are multiple ways to compute paths to the exit, with varying efficiency. Students lacking the "algorithmic" way of thinking benefit from seeing how the others tackle the problem.

Problem 2: Card Game. Students play a very simple game of cards where players pick cards turn by turn from the ends of a row of cards. The opponent plays a greedy strategy that seems to work well, and the goal is to beat him·her.

This is really a problem I love and a short description is provided in Appendix A. The problem itself is very simple to understand, and students always end up bringing actual cards to the sessions to try out strategies and produce counter examples. They are confronted to their first non optimal greedy algorithm, and already struggle to prove the non-optimality as they have first to define what it means. They soon find an exponential algorithm that tries every possibility. The key is to let students find by themselves (by reading lectures notes) that many sub-problems are recomputed and they can use dynamic programming to go back to polynomial complexity.

[2] Again, this grade is not so important that student start to rely on the group to pass the course instead of on their individual learning.

[3] Note that the literature usually recommend *against* letting students choose their own group, which is also what I advise for L1-L2 students, as there is a strong tendency to form "bad" groups where unmotivated students all end up, but I found that at the M1 level it does not pose any problem, as students are more invested in their learning.

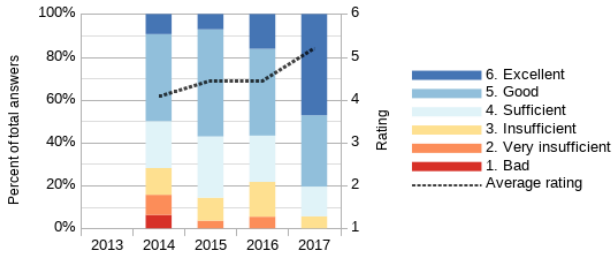


Figure 1: Evolution of the rating of the course by students

Problem 3: Traveling Salesman Problem. We disguise the TSP as a problem of drilling holes in a printed circuit board. Students are presented a very stupid algorithm which is not an approximation. Although there is no actual course on NP-completeness, students are introduced to the idea that some problems cannot be solved in polynomial time with our current knowledge. We then guide them into finding a 2-approximation using spanning trees. This is a drawback of PBL in this context: approximation algorithms is a very difficult topic and it is hard to find a problem that students can fully solve by themselves.

Problem 4: Morse Code. Students are asked to decipher a message in Morse code that was recorded without spaces. This last problem is shorter and meant to reconcile all that was learned during the semester: it includes fancy data structures (prefix trees), and the need to decide whether the problem is solvable using dynamic programming or needs to be approximated.

4 STUDENT'S FEEDBACK

When I initiated the switch to PBL in 2014, I received the help of a pedagogical councilor at the university, who had a lot of experience. He also proposed to conduct a first survey to get the feedback of students, which would help to improve the course. We kept using the same survey each following year, so we could track what was improving or not over the years. It should be noted that the feedback was asked *prior* to students passing the final exam and receiving the most important grade of the course. Sadly, I do not have data prior to switching to PBL.

4.1 The Survey

The survey is anonymous, and consists of 38 closed-ended questions and 11 open-ended questions. All close-ended questions are strongly worded, such as "The content is rich and interesting," and we propose four possible choices ("completely agree," "rather agree," "rather disagree" and "completely disagree") to encourage students to take position (not answering was considered "neutral"). Only one question, that asked student to globally rate the course, had six possible answers ranging from "excellent" (6) to "bad" (1). Results for this particular question are presented in Figure 1.

Questions are organized in categories. 6 questions on the global appreciation of the course: general content, session organization, material provided, and whether the choice of using PBL was well suited, allowed them to learn, and made them more responsible for their own learning. 7 questions on the group work: affinity toward working in group, group size, atmosphere, internal organization, workload repartition, quality of the group production,

Figure 2: Rating from students averaged by category

engagement of other members. 4 questions on the tutoring: understanding of the role of the tutor, listening/disponibility of tutor, technical skills, quality of supervision. 3 questions on the evaluation methods: group evaluation, quick tests, and indications for the final exam. 3 open-ended questions asking students to express in a few words the strengths of the course, its weaknesses, and if they had any suggestions to improve it. The rest of the questions concerned the individual problems (subject, difficulty, total time spend working on them, strengths and weaknesses). Results for these questions are presented averaged by category in Figure 2.

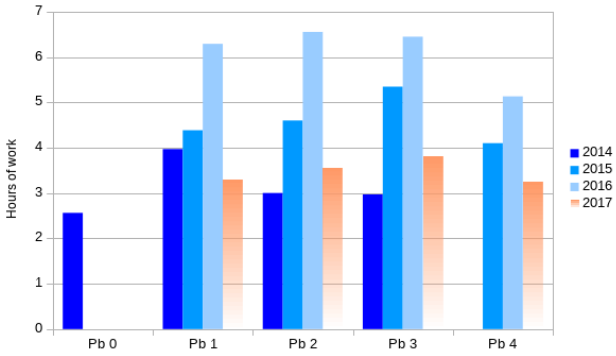
We managed each year to have the feedback of approximately 75% of students, i.e., from 28 to 38 people as the cohort increased slightly in size each year.

4.2 Interpretation of Results

Figure 1 shows the evolution of the global rating of the course by the students. The first year was a bit disappointing, as we spent a enormous amount of time trying to make this course a better one and converting everything to PBL. Still, about 70% of students had a rather positive view on the course, which was encouraging. The rating improved with the years, with a happy surprise this past year, with only two students rating the course as "insufficient," while nearly half of them thought it was "excellent."

Interestingly, the content of the course was very stable from 2015 to 2017, which is surprising when looking at Figure 1, which shows no improvement in 2016 but a large one in 2017. Considering Figure 2, which presents the results of the close-ended questions averaged by category for each year. We observe globally an increase in the rating ("completely disagree" is encoded as 1, while "completely agree" is coded by 4). However, it is even more clear that there is a 2016 "drop." Part of it can be explained by the unavoidable variability in students from year to year, but it is also probable that tutoring plays a role here.

The 2016 "drop". During all the study, I was tutoring half of the class, but in 2014, 2015, and 2016, there were three different tutors for the second half that I will call tutor A, tutor B and tutor C. While tutors A and B received training in PBL, tutor C specifically did not want to participate in such a training and hence received only minimal training from my part. In 2017, tutor C was again present for the second group, but this time had one year of experience plus half a day of meeting / debriefing on PBL in a medium group of 8-9 teachers. Since 2015, the content of the course was quite stable, so it



Note: 2017 is not comparable with others and Pb 4 spans only 4 sessions.

Figure 3: Reported hours of work per problem

is probable (as it could be expected) that the training of the teachers has an impact on the appreciation of the course by students.

Effects on group work. When looking at the detailed data (not present here due to size constraints), there is a correlation between the student's comprehensions of: the PBL method, the tutor's role (listening, supervision, guiding skills), and the group's internal organization and work repartition. All those were rated a bit low in 2016 and increased in 2017. This is again an indication that the tutor's presence and work has an effect on the group work, which in turns affects the students' global appreciation of the PBL method, and the course itself (even the *content* of the course was better appreciated in 2017). It should be noted that, even if the general rating for the "Tutor" category is quite high, students lack the ability to fully understand what is to be expected from a tutor, so it is best to factor these results with those of the "Group" category, where one can see the effects of "good" or "bad" tutoring.

General comments. Overall, problems are really appreciated by students, which is also reflected in the open-ended questions, where many comments stated it was very interesting to be confronted with "real-life" challenges that were engaging. Note that in 2015 we converted problem 0 to a single introductory session, and added problem 4 (which was not present in 2014).

The two worst-rated categories are "Group" and "Evaluation." For the evaluation, it shows that students do not feel prepared enough for the quick tests or the final exam. Indeed, compared to the other "classical" courses they have, and probably compared to the vast majority of other courses they had in their education, they feel they are not doing enough exercising with the PBL method. Even if the exam's grades have been stable, students needs reassurance, which we try to provide by including more exercises in the lecture notes and access to previous year's exams.

Finally, the group work is still the worst rated, even though students stated that they love working in group (85% reported a good atmosphere). The specific parts targeted are the internal organization and work repartition (only 55% positive). This shows the need for the tutor to diagnose dysfunctions in groups and propose solutions, which is probably the most difficult job of the tutor. See the conclusion for some ideas on how to improve.

Homework. We ideally would like students to devote one hour of personal work per hour of class, i.e., about 9h for a 6-session problem. Figure 3 presents the average number of hours that students reported working for each problem. Considering only the 2014–2016 period, the amount of personal work is increasing up to an average of a little more than 6h per problem, which indicates the engagement of students in their learning. This is also reflected by the fact that more than 90% of students agreed that PBL allowed them to think on their responsibility on their learning.

Sadly, we made a mistake in 2017 when switching to an electronic survey. While students were previously free to state exactly how many hours they worked, only limited ranges were given up to a maximum of 6h. It is highly likely that the results are very biased in 2017 (some reported more than 12h the previous years), probably even more due to anchoring effects.

5 EFFECTS ON STUDENTS AND TEACHERS

Students. By judging from past exams and exams done after the switch to PBL, I did not experience any significant change in the average ability of students to answer exam's questions. It should be noted that the final exam still has the same format as before, usually involving two very guided problems in dynamic programming and approximation algorithms. We do not test aptitudes specific to PBL such as the ability to work in group, to communicate better, and to have more autonomous thinking and generic problem-solving skills, (although the literature shows PBL support these [1]), as the master degree has specific requirements on purely algorithmic-related skills. It seems that students still get as much knowledge as previously, while getting orthogonal skills in communication and problem solving that will be useful to them in the professional world, and while enjoying the course a lot. From informal discussions with the students, this is now the preferred course of the master program.

Although it is sometimes difficult for students to work in groups, what I mainly observed was a strong cohesion between students with a desire to stay in the same group, and a lot of mutual assistance, with "better" students re-explaining difficult concepts to "weaker" students, which for me is a success. Sadly, not all groups worked perfectly. Here are some quotes from the students' survey that reflect the above points:

> "Finding a solution on our own, allowing us to be creative is a very good thing in my opinion." *(question on positive aspects)*

> "The problem was that students have different levels in the algorithms depending on their background. So students with good background can go on with the [problem] while other students are struggling."

> "It's very hands on, which allowed the class to be fun. I learned as much from my own teammates than from the material."

Teachers. PBL changes also a lot of things from the teacher's point of view. While student's performance and engagement is the main goal of a course, these are positively or negatively affected by the teacher's state of mind. More generally, it is also important for me to feel at the right place when I'm working. Teaching using PBL (in this master course as well as in other courses) has brought be and my co-tutors closer to my students, removing the feeling of not really knowing my students. It used to take nearly a full semester to start knowing everyone in a small group of 20-30, now it takes two to three weeks. This is probably the most satisfying aspect of

PBL to me: being a tutor that interacts with its students instead of a distant teacher.

Not to say that everything is perfect. As mentioned previously, I worked with three different co-tutors for the four years of PBL. The first year, we did a great amount of work with tutor A to set things up and we both enjoyed a lot the tutoring. However, tutor A decided to quit the university at the end of the year, feeling there was too much pressure as a researcher to be able to do its teaching job right. Although this is not directly related to PBL, it is true that creating a PBL course from scratch takes an enormous amount of time, much more than a "classical" course from what I experienced, and this can feel overwhelming. I sometimes personally felt close to burnout. So I would advise to prepare the switch to PBL maybe up to a year in advance.

Tutor B then worked with me for a year. While he also enjoyed the course, he felt he was not competent enough in algorithms to be a good tutor. It is certainly an aspect of tutoring that is very stressful for novice tutors, compared to a classical exercising session. While it is possible to be well prepared for the latter, PBL needs a very reactive tutor as free students always come up with a variety of different approaches for the same problem. Even after four years, there are always groups who propose a different way of solving the problems, and it requires a lot of confidence from the tutor to be able to validate or invalidate the proposed solutions. It is in particular crucial to discern if learning will take place if students continue their lead, of if we need to guide them back to the tracks. I find this extremely interesting, as this aspect of PBL really allows teachers to better understand the reasoning and misconceptions of students, but we have a greater responsibility at the same time.

Lastly, tutor C has been my fellow co-tutor for two years now, and is quite happy about it, although not yet fully convinced by the method, and would gladly step back "just a bit" towards the "classical" way of teaching, with more exercises and more formal lectures. This is a paradox I have often observed in fellow teachers who did not actively search for an alternative way of teaching: although the evidence that PBL is great for teaching seems overwhelming (to me at least), the changes it requires to be a tutor is so great that teachers are put really out of our "comfort zone" and it requires much mental effort to *not* go back.

6 CONCLUSION

Making the jump to Problem-Based Learning allowed me to be more in contact with students, know them better and guide them in their learning. Although it seems we can teach less content, it is not so true for the amount of *meaningful* content, i.e., content that students are really likely to remember and make theirs. Students are more engaged in their learning: they feel more responsible and also commit more through personal work. By making them solve actual problems, and letting them be confronted to the difficulty of the act of solving, students feel more connected to the reality and view the course as more useful. It is my hope that whatever is gained that way remains in memory for longer, as it has been shown in the literature [7], and is reflected by this student's comment on the strengths of the course:

> "Learning by yourself, so getting experience, which in my opinion makes you keep the knowledge further than just the exam."

Moreover, students in this course develop transversal skills in communication, group work, and mutual assistance. As a result, it allowed me to satisfactorily teach a class of mixed students having with very different backgrounds and levels of knowledge in algorithms.

As biased as I may be towards PBL, there are still difficulties that need to be assessed. Firstly, students are often more stressed by the exams as they feel they are not prepared enough, because they are solving "only four problems" in one semester. A way to improve this is to also give "classical" exercises on the side, or make formative online assessments, for instance using multiple choice questions. Secondly, working in group is not always easy, and a group that highly dysfunctions would really hinders the learnings of everyone in the group. Hopefully I was always able to soothe things out for now. For instance, it is particularly important with such a setting that every member does its personal work. This may be the most difficult facet of the tutoring, hence it is especially important that teachers receive training before engaging in PBL.

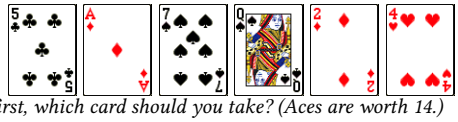
I will finish by a citation of M. Legrand, talking about lectures:

> "One problem with classical school is that it gives answers to questions we didn't ask and techniques to solve problems we don't have…" [4]

This is where resides the power of Problem-Based Learning: it starts by giving problems, and then students have questions...

A PROBLEM EXAMPLE: HOLD'EM N00BS

This is a 2-player game. There is a series of n cards lying on the table face up arranged in a line. Players take turns to take cards one at a time, but only the rightmost or the leftmost card can be picked of the line. When the last card is taken by a player the game is finished. The highest sum of points of all cards taken wins the game.



You play first, which card should you take? (Aces are worth 14.)

ACKNOWLEDGMENTS

Thanks to Christophe Durand from the university's pedagogical service, for his invaluable help in making the switch to PBL.

REFERENCES

[1] Samuel B. Fee and Amanda M. Holland-Minkley. 2010. Teaching computer science through problems, not solutions. *Computer Science Education* 20, 2 (2010), 129–144.
[2] Judy Kay, Michael Barg, Alan Fekete, Tony Greening, Owen Hollands, Jeffrey H. Kingston, and Kate Crawford. 2000. Problem-Based Learning for Foundation Computer Science Courses. *Computer Science Education* 10, 2 (2000), 109–128.
[3] Päivi Kinnunen and Lauri Malmi. 2005. Problems in Problem-Based Learning – Experiences, Analysis and Lessons Learned on an Introductory Programming Course. *Informatics in Education* (2005).
[4] Legrand Marc. 2017. Le débat scientifique en classe. (2017). IXème Colloque Questions de Pédagogies dans l'Enseignement Supérieur.
[5] Esko Nuutila, Seppo Törmä, and Lauri Malmi. 2005. PBL and Computer Programming The Seven Steps Method with Adaptations. *Computer Science Education* 15, 2 (2005), 123–142.
[6] H. G. Schmidt. 1983. Problem-based learning: rationale and description. *Medical Education* 17, 1 (1983), 11–16.
[7] Shin, Haynes, and Johnston. 1993. Effect of problem-based, self-directed undergraduate education on life-long learning. *CMAJ: Canadian Medical Association Journal* 148, 6 (1993), 969–76.